UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/902,133	07/11/2001	Seiji Hayashida	211316US2	7963

22850 7590 05/06/2004

OBLON, SPIVAK, MCCLELLAND, MAIER & NEUSTADT, P.C.
1940 DUKE STREET
ALEXANDRIA, VA 22314

EXAMINER	
RAMPURIA, SATISH	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 05/06/2004

7

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/902,133	HAYASHIDA, SEIJI	
	Examiner	Art Unit	
	Satish S. Rampuria	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 July 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *07/11/2001*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the application filed on 07/11/2001.

2. Claims 1-20 are pending.

Priority

3. Acknowledgment is made of applicant's claim for foreign priority under 35 U.S.C. 119(a)-(d). The certified copies have been received on July 11, 2001.

Specification

4. The disclosure is objected to because of the following informalities: On page 1, line 24 word "the" after "instruction," appears to be redundant and on page 3, line 12, "instrincsic" should be "intrinsic".

Appropriate correction is required.

Information Disclosure Statement

5. An initialed and dated copy of Applicant's IDS form 1449, Paper No. 05, is attached to the instant Office action.

Claim Rejections - 35 USC § 112, second paragraph

6. Claims 17, 19, and 20 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Clarification and/or correction are required.

Regarding, claim 17, the body of the claim does not appear to limit the program recited in the preamble, rather appears to recite steps of a method which would have been performed upon execution of the program. Therefore it is unclear whether the claim recites a program or a

method. For purposes of prior art search, the claim is interpreted as a compiling process including the steps as recited.

Claims 19 and 20 have similar limitation to those in claim 17 with respect to the program and are jected for the same reason.

The rejection of the base claim is necessarily incorporated into the dependent claims.

Claim Rejections - 35 USC § 101 Utility

7. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

8. Claims 1-7 and 17-20 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The claims are non-statutory because they recite components of a compiler, representing functional descriptive material without a computer readable medium. Claims 1-7 and 17-20 thus amounts to only abstract idea and are nonstatutory.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,083,282 to Caron et al., hereinafter called Caron, in view of US Patent No. 6,247,174 to Santhanam et al., hereinafter called Santhanam.

Per claims 1 and 2:

Caron discloses:

- A compiler for generating object code from an input source program (col. 1, lines 21-23 "high level language... source code are then translated or compiled into the coded instructions executable by the computer")
- a character string interpreter which divides instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "compiling process 70... performed on source code to separate the source code into various lexical constructs or tokens... programming language, including identifiers, keywords, operators, literals, and comments")
- a syntax analyzer which analyzes syntax of said tokens (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like"), and makes a judgment (col. 9, lines 35-37 "compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code")

Caron does not explicitly disclose whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens, an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information, and a code generator which develops an instruction that calls an

intrinsic function within said source program by referring to said intrinsic function information, and which converts said developed source program either to machine language or to an intermediate code.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct") and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") and a code generator (col. 6, lines 30 "Code generator 105") which develops an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s"), and which converts said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 3:

The rejection of claim 1 is incorporated, and further, Caron doe not explicitly discloses intrinsics function definition includes a dummy argument type and identification name.

However, Santhanam in an analogous computer system discloses intrinsics function definition includes a dummy argument type and identification name (col. 4, lines 10-13 "The table contains one entry for each intrinsic... name... data type... argument... value").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of containing intrinsics function information as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to include the intrinsics function definition to improve the run time performance of the library function invocation as suggested by Santhanam (col. 2, lines 60-65).

Per claim 4 and 5:

Caron discloses:

- A compiler for generating object code from an input source code program (col. 1, lines 21-23 "high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer"), comprising:
- a character string interpreter which divides instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments")
- a syntax analyzer, which analyzes syntax of said tokens (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures, such as declaration statements, loop statements, expressions, and the like")

Caron does not explicitly disclose whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens, an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information, and a code generator which develops an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information, and which converts said developed source program either to machine language or to an intermediate code.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct") and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") and a code generator (col. 6, lines 30 "Code generator 105") which develops an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s"), and which converts said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The

modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 6:

The rejection of claim 5 is incorporated, and further, Caron doe not explicitly discloses intrinsics function definition includes a dummy argument type and identification name.

However, Santhanam in an analogous computer system discloses intrinsics function definition includes a dummy argument type and identification name (col. 4, lines 10-13 "The table contains one entry for each intrinsic... name... data type... argument... value").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of containing intrinsics function information as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to include the intrinsics function definition to improve the run time performance of the library function invocation as suggested by Santhanam (col. 2, lines 60-65).

Per claim 7:

Caron discloses:

- A compiler for generating object code from an input source program (col. 1, lines 21-23 "high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer"), comprising:
- a character string interpreter which divides instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments")
- a syntax analyzer which analyzes syntax of said tokens (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures, such as declaration statements, loop statements, expressions, and the like")
- wherein said intrinsics function information includes a function declaration statement, to which is added a prescribed identifier indicating an intrinsics function, dummy argument information, and said instruction attribute information.

Caron does not explicitly disclose whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens, an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information, and a code generator which develops an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information,

and which converts said developed source program either to machine language or to an intermediate code and intrinsics function definition includes a dummy argument type and identification name.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates that code containing embedded machine instructions is... correct... when... incorporated... source code 101 in the same way... a front end verifies.... function invocation...") and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") and a code generator (col. 6, lines 30 "Code generator 105") which develops an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s"), and which converts said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106") and intrinsics function definition includes a dummy argument

type and identification name (col. 4, lines 10-13 "The table contains one entry for each intrinsic... name... data type... argument... value").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claims 8, 9, 10, and 11:

Caron discloses:

- A method for compiling which generates object code from an input source program (col. 1, lines 21-23 "high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer"), comprising:
- dividing instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments");

- analyzing the tokens and detecting from a combination of said tokens a declaration of a start of coding with regard to said intrinsics function (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures, such as declaration statements, loop statements, expressions, and the like")

Caron does not explicitly disclose storing a definition of an intrinsics function into an intrinsics function information database; storing instruction attribute information characterizing an instruction coded by an intrinsics function into said intrinsics function information database; developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information database, and converting said developed source program either to machine language or to intermediate code.

However, Santhanam in an analogous computer system discloses storing a definition of an intrinsics function into an intrinsics function information database (col. 14, lines 54-57 "information ... identity... of... intrinsic... described... is stored in the same data structures described in A") and storing instruction attribute information characterizing an instruction coded by an intrinsics function into said intrinsics function information database (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") and developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information database (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s"), and converting said developed source program either to

machine language or to intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of storing the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to store the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Claim 12 is the computer program product claim corresponding to system claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 13 is the computer program product claim corresponding to method claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Claim 14 is the computer program product claim corresponding to method claim 1and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 15 is the computer program product claim corresponding to method claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Claim 16 is the computer program product claim corresponding to method claims 1 and 3 respectively, and rejected under the same rational set forth in connection with the rejection of claim 1 and 3 respectively, above.

Per claims 17 and 18:

Caron discloses:

- processing for character string interpretation so as to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "compiling process 70... performed on source code to separate the source code into various lexical constructs or tokens... programming language, including identifiers, keywords, operators, literals, and comments")
- processing for analyzing said tokens, and performing syntax analysis (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like") so as to judge (col. 9, lines 35-37 "compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code")

Caron does not explicitly disclose whether or not a combination of the tokens has an intrinsics function definition and a definition of instruction attribute information characterizing an instruction coded by said intrinsics function; processing for storing said intrinsics function definition and intrinsics function information as intrinsics function information; processing for

developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information; and processing for generating code that converts said developed source program either to machine language or to intermediate code.

However, Santhanam in an analogous computer system discloses, whether or not a combination of the tokens has an intrinsics function definition and a definition of instruction attribute information characterizing an instruction coded by said intrinsics function (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct") and processing for storing said intrinsics function definition and intrinsics function information as intrinsics function information(col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") processing for developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s") and processing for generating code (col. 6, lines 30 "Code generator 105") that converts said developed source program either to machine language or to intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 19:

Caron discloses:

- processing for character string interpretation so as to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "compiling process 70... performed on source code to separate the source code into various lexical constructs or tokens... programming language, including identifiers, keywords, operators, literals, and comments")
- processing for analyzing the syntax of said tokens (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like") so as to judge (col. 9, lines 35-37 "compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code")

Caron does not explicitly disclose processing for storing an intrinsics function definition and attribute information characterizing an instruction coded by said intrinsics function as intrinsics function information; processing for accessing said intrinsics function information; processing for developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information; and processing for generating code that converts said developed source program either to machine language or to intermediate code.

However, Santhanam in an analogous computer system discloses, processing for storing an intrinsics function definition and attribute information characterizing an instruction coded by said intrinsics function as intrinsics function information (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct") processing for accessing said intrinsics function information (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") processing for developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information; and (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s") and processing for generating code (col. 6, lines 30 "Code generator 105") that converts said developed source program either to machine language or to intermediate code (col. 6, lines 30-33

"Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 20:

Caron discloses:

- processing for character string interpretation so as to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 "In the syntax of the programming language used... names (also known as identifiers) are strings of characters") and col. 9, lines 21-24 "compiling process 70... performed on source code to separate the source code into various lexical constructs or tokens... programming language, including identifiers, keywords, operators, literals, and comments")
- processing for analyzing the syntax of said tokens (col. 9, lines 24-27 "Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like") so as to judge (col. 9, lines 35-37

"compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code")

Caron does not explicitly disclose processing for storing an intrinsics function definition and attribute information characterizing an instruction coded by said intrinsics function as intrinsics function information; processing for accessing said intrinsics function information; processing for developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information; and processing for generating code that converts said developed source program either to machine language or to intermediate code and wherein said intrinsics function information is made up of a function declaration statement to which is added a prescribed identifier indicating an intrinsics function, dummy argument information, and said instruction attribute information.

However, Santhanam in an analogous computer system discloses, processing for storing an intrinsics function definition and attribute information characterizing an instruction coded by said intrinsics function as intrinsics function information (col. 12, lines 46-53 "Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct") processing for accessing said intrinsics function information (col. 4, lines 10-15 "The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction") processing

for developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information; and (col. 7, lines 1-3 "programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s") and processing for generating code (col. 6, lines 30 "Code generator 105") that converts said developed source program either to machine language or to intermediate code (col. 6, lines 30-33 "Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106") and wherein said intrinsics function information is made up of a function declaration statement to which is added a prescribed identifier indicating an intrinsics function, dummy argument information, and said instruction attribute information (col. 4, lines 10-13 "The table contains one entry for each intrinsic... name... data type... argument... value").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Conclusion

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

The following patent is cited to further show the state of the art with respect to generating object code from an input source program/file.

US Patent No. 5,606,697 to Ono

US Patent No. 6,629,313 to Rowe et al.

US Patent No. 5,832,273 to Mizuse

Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Satish S. Rampuria** whose telephone number is **703-305-8891**. The examiner can normally be reached on **8:30 am to 5:00 pm**.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Kakali Chaki** can be reached on **(703) 305-9662**. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria

Patent Examiner

Art Unit 2124

05/03/2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100